2012 12th IEEE-RAS International Conference on Humanoid Robots
Nov.29-Dec.1, 2012. Business Innovation Center Osaka, Japan

Real-time Continuous Collision Detection for Mobile Manipulators – A General Approach

Holger Täubig
Cyber-Physical Systems,
DFKI, Bremen, Germany
holger.taeubig@dfki.de

Berthold Bäuml
Institute of Robotics and Mechatronics,
DLR, 82234 Wessling, Germany
berthold.baeuml@dlr.de

Udo Frese
Cyber-Physical Systems,
DFKI, Bremen, Germany
udo.frese@dfki.de

***Abstract*—We present a general real-time continuous collision detection algorithm for arbitrary systems of moving bodies connected to each other in a kinematic tree of joints. Here "joint" as a general term refers to the measured relative motion between two bodies, which may be physically connected or not. We provide a basic set of joints covering revolute and prismatic joints, vehicle motion, and 3d positioning which is sufficient for many applications in particular those involved with mobile manipulators, e.g. industrial and humanoid robots, intelligent transportation systems, or equipment at construction sites. Each joint implementation either operates on a motion bound (an interval covering the braking distance) or an uncertainty bound (measurement error) and computes a volume that spatially bounds the effect of that motion or uncertainty. Aggregating all joints in a kinematic tree then yields a conservative continuous collision detection for the complex and uncertain motion of the whole system.**

We further present an augmented reality visualization that overlays the collision volumes into the live image of a camera, which can be used to validate the collision model before bringing a system into service.

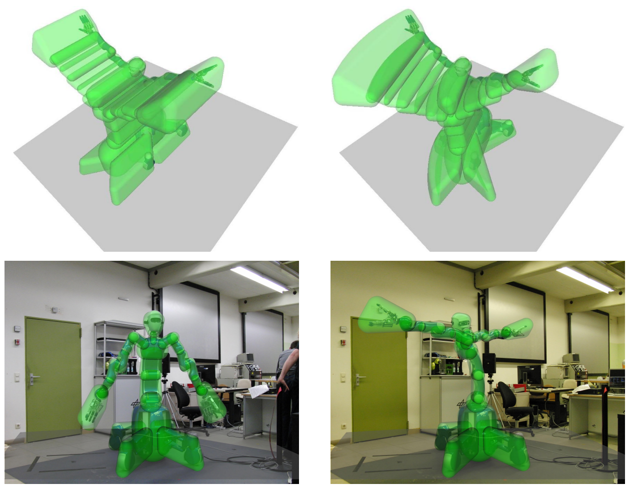
Fig. 1. Swept volumes of a humanoid robot (DLR's Justin, [1]) induced by linear *(top left)* and rotational *(top right)* motion of its mobile platform. *bottom:* Visualization of the collision volumes in a camera image, e.g., for a verification of the collision model.

I. Introduction

Mobile manipulators, i.e., manipulators mounted on a mobile platform (Fig. 1, 2), provide a wide range of applications covering industrial and humanoid robots, intelligent transportation systems, and even equipment at construction sites (e.g., cranes, lifting platforms). Certainly, all of these applications have a demand for a collision prevention system, either for monetary or for safety reasons. Further, they involve motion and show considerable measurement errors as a lot of real world applications do, so there is the need for a geometric tool that provides real-time collision detection for mobile manipulators under consideration of (braking) motion and significant uncertainty. In this paper, we extend our approach from [2] for joint manipulators into a seamless integrated vehicle-manipulator continuous collision detection system.

Collision detection as the prerequisite of collision avoidance has been subject of research for decades now. In the past, there were separated 2d and 3d approaches: 2d collision avoidance involved vehicles (intelligent transportation systems, autonomous guided vehicles (AGVs), autonomous wheelchairs) moving in a plane; 3d collision avoidance originally involved robot manipulators, i.e. kinematics built from revolute and prismatic joints, but without wheeled motion. For mobile manipulators one has to overcome this separation. Our approach is to provide a general set of joint types that can be assembled in a kinematic tree, which describes the (braking) motion and uncertainty of both robot and environment. The motion effect of the kinematic tree is computed conservatively for a set of bounding volumes representing all moving and static bodies connected to the kinematic tree. The result are swept volumes (Fig. 1), i.e. the volumes potentially covered by each body due to its overall motion and uncertainty. Finally, the swept volumes of all body-pairs are checked for collisions by computing distances.

Research supported by the DFG under Grant No. FR2620/1-1 (BCATCH).

The main contribution of this paper is to provide computations for a general set of joint types covering revolute and prismatic joints, vehicle motion, and 3d positioning. In particular, we derive a vehicle joint from previous work [3] in 2d collision detection. We additionally provide multiple versions of each joint type, which allows for a trade-off between accuracy and computational effort. This joint set is sufficient for many applications in particular mobile ma-

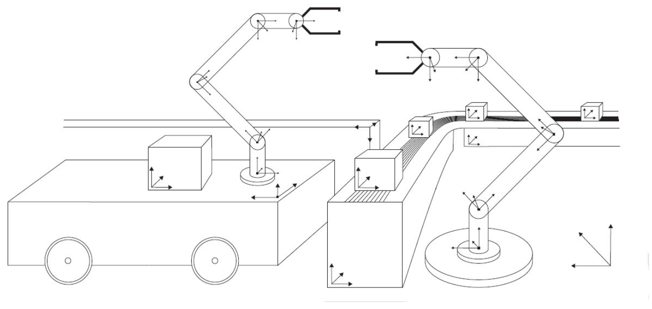

Fig. 2. General industrial application scenario motivating our approach.

nipulators and can be easily extended for application specific joints. The overall result is a general continuous self collision detection for multi-body systems without parallel kinematics. Conservative, continuous collision detection comes with propagating a body's volume along its kinematic chain by taking the effect of one joint after the other into account; computational efficiency comes with the use of so called sphere swept convex hulls (SSCH, Sec. III, (1)) and the distance update scheme from [2]. If this scheme is used, the computation time of the algorithm is bounded, hence, the algorithm is real-time capable. Experiments, in simulation and on the real system, are performed with DLR's humanoid Justin [1] (Fig. 1).

An additional contribution of this paper is the augmented reality visualization (Fig. 1). It allows to validate the collision model, which is important for safety and also demanded by standards such as IEC 61508.

The paper is organized as follows: After related work in Sec. II, we present the algorithm in Sec. III, followed by visualization (Sec. IV) and experiments (Sec. V).

II. Related Work

Besides in robotics, also in computer graphics the problem of collision detection has been thoroughly studied (see books [4], [5] for an extensive and intelligible overview). Computer graphics usually considers bodies arbitrarily moving in space, whereas robotics focuses on the kinematic tree that moves these bodies. So, computer graphics starts from an arbitrary pose (resp. motion) for each body, whereas robotics considers how it is constituted by a kinematic chain of motion primitives. This kinematic chain represents the robot's technical construction from potentially different types of joints. This is also our particular focus in contrary to libraries such as PQP [6], SWIFT [7], V-Clip [8], SOLID [5], or CCD [9].

A further demand for collision detection and visualization of mobile manipulators arose in the area of automation in construction [10][11] usually for vehicles with a 3-DOF manipulator. Kim [10] explicitly mentions that but does not handle the vehicle. Hwang's approach [11] only applies to simple bar-shaped bodies like booms of tower cranes. However, construction sites often involve vehicles and complex-shaped objects where we think our more general approach is useful.

Established 2d algorithms like the dynamic window approach [12] or the nearness diagram [13] provide continuous collision detection for vehicles, but are not easily transferable to 3d. A typical motion model in 2d is circular motion. Some approaches compute the area potentially touched by the vehicle and stop if an obstacle is inside. Often the area is represented as a grid which does not scale well to 3d. Therefore, in [3] we proposed a 2d SSCH representation, which we will extend here to 3d as the vehicle joint. Other approaches such as collision cones [14], [15] and velocity obstacles [16], [17] work in 3d but only for linear motion.

Collision detection in 3d originally considered discrete configurations [5], [6], [7], [8] as opposed to time intervals for computational reasons. Recent approaches considered continuous collision detection [9] but simplified the overall motions of each body instead of handling the kinematic structure as our approach does.

Further, [18] use s-topes, a generalization of SSCHs. This representation is more general but makes distance computation more complex than for SSCHs. In [19] the authors propose to cover the trajectory induced by a revolute joint by the convex hull of multiple points, as we do. However, we are strictly conservative by adding a conservative "error bound" as a buffer radius and generalize this idea to different kinds of joints here.

III. Algorithm

We consider a system of *bodies* moving in 3d space that are connected via a number of *abstract joints*. For each Body B_i there is a body fixed frame C_i attached to B_i. An abstract joint is a time-varying transformation between two frames, resp. bodies, thus it provides their relative motion over time. It is abstract in the sense that it can represent a real physical joint like a revolute or prismatic joint, thus a physical connection between two bodies, or any other relative motion such as the path of a vehicle with respect to a world frame for example. A joint can even represent the measurement of a relative pose in terms of 3d rotation and translation if that is available within the system. In all cases a joint defines the relative motion of two bodies. The whole system's motion is established by a tree of joints, which we call the kinematic tree and which defines the system's forward kinematics. In that, each joint is of a certain type and depends on values from the configuration vector q, e.g. a joint angle.

We use sphere swept convex hulls (SSCH) for volume representation, which we introduced in [2]. A SSCH represents a volume by a finite set of points $[p_k]_{k=1}^n$ and a radius r

$$V\big(r;\ [p_k]_{k=1}^n\big) = \operatorname{conv}\{[p_k]_{k=1}^n\} + \big\{b \in \mathbb{R}^3 \bigm| |b| \le r\big\}, \quad (1)$$

where conv is the convex hull of a given set of points. So each volume is the Minkowski-sum of a convex polyhedron given by a set of points, and a ball of radius r.

Denoting static volumes in frame C_j by $V_j \subset \mathbb{R}^3$, the volume V_i^i of a body B_i in C_i is given as an SSCH bounding volume

$$V\big(r_i;\ [p_k^i]_{k=1}^{n_i}\big) \supset V_i^i. \quad (2)$$

The V_i^i together are the collision model, [2] describes how it is obtained conservatively from a detailed CAD model. The proposed algorithm will, except for visualization, only operate on points $[p_k]_{k=1}^n$ and radii r but never has to compute the volume $V\big(r_i;\ [p_k^i]_{k=1}^{n_i}\big)$ or the convex hull involved explicitly.

A. Joint Specification

Beforehand some notation: A transformation $T_{j\leftarrow i}$ is a matrix mapping of C_i-coordinates to C_j-coordinates (either static or at a single point in time). $\mathcal{Q} = \{q(\lambda)|\lambda \in [0,1]\}$ represents the motion in the configuration space. Now, consider joint J_i, which is the joint body B_i and frame C_i move with. Let's denote its parent joint in the kinematic tree with J_j and formulate the effect of joint J_i as its motion $\mathcal{T}_{j\leftarrow i} = \{T_{j\leftarrow i}(q)|q \in \mathcal{Q}\}$ represented in the parent frame C_j. Consequently, the swept volume $\mathcal{V}_j$ of a volume V_i that moves with J_i is

$$\mathcal{V}_j = \mathcal{T}_{j\leftarrow i} \cdot V_i = \{T_{j\leftarrow i}(q) \cdot p|q \in \mathcal{Q}, p \in V_i\}\,. \tag{3}$$

The concrete motion of J_i

$$\mathcal{T}_{j\leftarrow i} = T_{j\leftarrow i*} \cdot \mathcal{T}_{i*\leftarrow i} \quad \text{with } \mathcal{T}_{i*\leftarrow i} = \{X(a,q)|q \in \mathcal{Q}\} \tag{4}$$

further depends on joint type X, a type specific parameter vector a, and a fixed transformation $T_{j\leftarrow i*}$ representing the location and orientation of J_i in the parent frame C_j. All of these parameters are given in the kinematic tree. $\mathcal{T}_{i*\leftarrow i}$ is the pure, location-independent motion of J_i, e.g., a rotation around the origin for a revolute joint. Frame C_i moves while the intermediate frame C_{i*} does not. In the null configuration $q = 0$ both coincide. The joint's motion model X defines the transformation between frames C_i and C_{i*} depending on q. It is a specific function X for each joint type, e.g. revolute, prismatic joints or others, which takes as input parameters a, e.g. an axis, and a configuration q, e.g. joint angles, and returns a transformation matrix from C_i into C_{i*}.

To add an implementation for joint type X to our framework means to provide a conservative approximation of the effect of X for a set of configurations $\mathcal{Q}$ but only on a single point p moving with the joint

$$\mathcal{V}\big(r;\ [p^1,\dots,p^L]\big) \supset \mathcal{T}_{i*\leftarrow i} \cdot p. \tag{5}$$

This is done by defining a function OP_X that returns points $p^1,\dots,p^L$ and a radius r of an SSCH that fulfills (5) for the function X modelling the joint type.

$$OP_X(a,\mathcal{Q},p) := \big(r;\ [p^1,\dots,p^L]\big) \tag{6}$$

Given a correct OP_X the framework automatically applies OP_X to all points of the input volume $V\big(r;\ [p_k]_{k=1}^n\big)$ which yields a conservative approximation of its swept volume constituted by J_i:

$$\mathcal{V}\big(r + \max_k r_k;\ [[T_{j\leftarrow i*} \cdot p_k^l]_{l=1}^L]_{k=1}^n\big) \supset \mathcal{T}_{j\leftarrow i} \cdot V\big(r;\ [p_k]_{k=1}^n\big), \tag{7}$$

with $\big(r_k;\ [p_k^l]_{l=1}^L\big) = OP_X(a,\mathcal{Q},p_k)$. This is a specific property of the SSCH representation exploited by our framework. A proof of (7) can be found in [2].

B. Swept Volume and Distance Computation Algorithm

We use the overall algorithm we proposed in [2]:

1) **Compute all joint intervals** $\mathcal{Q} = [q^0; q^1]$, such that when the robot starts braking in the next cycle, it will stop within this interval. The intervals are based on joint angles, joint angle velocities, latency, and worst-case deceleration, as well as joint angle uncertainties.
2) **Compute swept volumes $\mathcal{V}_k^i$ of all bodies B_i in all joints J_k from the body down to the world frame by successively including the sweeping effect of one joint J_k after the other.** The swept volumes are represented in coordinates of the corresponding joint-frame C_k of joint J_k.[1]
3) **For each body pair** (B_i, B_j)

   Compute the distance of $\mathcal{V}_k^i$ and $\mathcal{V}_k^j$ in the first common joint-frame C_k on the sequences of joints from B_i and B_j down to the world frame.
4) **Stop the robot if any of the distances from 3 is zero.**[2]

In addition to this algorithm we also apply the distance update scheme we introduced in [2], thus the real-time algorithm that refines step 3. Due to the different nature of the joint types to be modelled later on, $\mathcal{Q}$ in step 1 may be represented by a nominal value plus an uncertainty bound instead of an interval for some kinds of joints. Step 2 applies the effect of one joint after another. It conservatively bounds the swept volume of a chain of joints $k_1,\dots,k_{m-1}$

$$\mathcal{V}_{k_m}^i \subset \prod_{j=m-1}^{1} \mathcal{T}_{k_{j+1}\leftarrow k_j}(\mathcal{Q}_{k_j}) \cdot V_{k_1}^i \tag{8}$$

by considering all combinations of joint configurations $\mathcal{Q} = \mathcal{Q}_{k_1} \times \ldots \times \mathcal{Q}_{k_{m-1}}$. In (8) k_1 is the body-fix frame of $\mathcal{V}^i$ and rightmost in the product and k_m is the frame for which the swept volume of $\mathcal{V}^i$ is desired and leftmost.

C. The Set of Joint Types

In what follows, we provide implementations for some basic joint types. For some types we show several alternative implementations, allowing to trade-off between accuracy and computation time, the latter mainly determined by the number of points returned by a solution. Any of the provided solutions are conservative. For a point p and $\mathcal{Q} = [q_0, q_1]$ we denote the result of the joint model at time $\lambda \in [0,1]$ by

$$p^\lambda = X(a, q(\lambda)) \cdot p \quad \text{with } q(\lambda) = (1-\lambda)q_0 + \lambda q_1. \tag{9}$$

1) Revolute Joint: A 1-DOF joint that provides single axis rotation. The centre of the rotation $X_{rev}(a,\alpha)$ is the origin and the parameter a is the rotation axis. The configuration $q = \alpha$ of a revolute joint is a joint angle. The operation input $\mathcal{Q} = [\alpha_0, \alpha_1]$ is a joint angle interval that bounds motion (e.g. braking distance) and uncertainty of α. We assume $|\alpha_1 - \alpha_0| < \pi$. The point p moves on a circular arc within angle

[1] $\mathcal{V}_k^i$ is represented in C_k but does not include the sweeping effect of J_k (cf. notation in Sec. III-A).

[2] Alternatively, less or equal a configured *safety distance*.

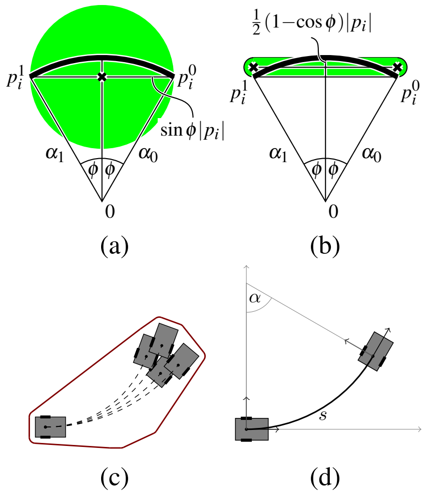

Fig. 3. (a) and (b) Revolute joint computation: bounds of a circular arc by an SSCH with 1 and 2 points. (c) Vehicle joint computation. (d) parametrization of vehicle braking distance as $(s, \alpha)^T$.

α_0 and α_1 around axis a. Operations may cover the arc by an SSCH of 1 or 2 generating points (more are theoretically possible):

$$\mathrm{Circ}_1\,(a, [\alpha_0, \alpha_1], p) := \big(\sin|\phi|\ \|p\|;\ [\tfrac{1}{2}(p^0 + p^1)]\big) \tag{10}$$

$$\mathrm{Circ}_2\,(a, [\alpha_0, \alpha_1], p) := \big(f\|p\|;\ [\,p^0 + f p^{1/2},\ p^1 + f p^{1/2}]\big) \tag{11}$$

$$\text{with } \phi = \tfrac{\alpha_1 - \alpha_0}{2} \text{ and } f = \tfrac{1-\cos\phi}{2}.$$

Fig. 3(a)-(b) show a 2d visualization of Circ_1 and Circ_2. Circ_1 is faster while Circ_2 is more accurate.

2) Prismatic Joint: A 1-DOF joint that provides single axis sliding. $X_{pris}(a, d)$ is a translation; parameter a is the translation axis. The configuration is a slider position $q = d$ denoting the translation distance along a. The operation input $\mathcal{Q} = [d_0, d_1]$ is an interval that bounds motion (e.g. braking distance) and uncertainty of the slider position d. The point p moves along a line segment of length $d_1 - d_0$ which can be covered by an SSCH of 1 or 2 generating points. The first just takes the middle and adds a radius, the second takes both endpoints and is exact.

$$\mathrm{Trans}_1\,(a, [d_0, d_1], p) := \big(\tfrac{1}{2}|d_1 - d_0|;\ [\tfrac{1}{2}(p^0 + p^1)]\big) \tag{12}$$

$$\mathrm{Trans}_2\,(a, [d_0, d_1], p) := \big(0;\ [\,p^0,\ p^1]\big). \tag{13}$$

3) Orientation Joint: A 3-DOF joint that allows for an arbitrary 3d rotation with rotation centre at the origin. Its purpose is to incorporate relative orientation measurements, thus it usually does not represent a physical connection of bodies. Compared to a revolute joint, a 3d rotation joint takes the rotation axis from the configuration vector instead of having a fixed parameter. However, apart from the source of the axis the models are equivalent $X_{rot3}(\_, [a, \alpha]^T) = X_{rev}(a, \alpha)$. We use an axis-angle representation $q = [a, \alpha]^T$ as the configuration.

Instead of some multi-dimensional interval, the orientation joint takes a single valued orientation error bound, which is the maximum possible angle β between measured and true orientation. Thus, the operation input $[a, \alpha, \beta]$ represents

$$\mathcal{Q} = \left\{ \left[\begin{smallmatrix} a' \\ \alpha' \end{smallmatrix}\right] \;\middle|\; \left|\mathrm{angle}\big(X_{rot3}(\_, \left[\begin{smallmatrix} a \\ \alpha \end{smallmatrix}\right]) \cdot X_{rot3}^{-1}(\_, \left[\begin{smallmatrix} a' \\ \alpha' \end{smallmatrix}\right])\big)\right| \le \beta \right\}. \tag{14}$$

The resulting spatial effect of $\mathcal{Q}$ for a point p is the intersection of a conical sector of angle β with the surface of a sphere of radius $\|p\|$, thus a circular sphere sector. The following operations using SSCHs of 1 or 4 generating points are 3d extensions of the circular arc approximations in (10) and (11)

$$\mathrm{Rot3D}_1\,(\_, [a, \alpha, \beta], p) := \big(\sqrt{2 - 2\cos(\beta)}\ \|p\|;\ [p^*]\big) \tag{15}$$

$$\mathrm{Rot3D}_4\,(\_, [a, \alpha, \beta], p) := \big(f_1\|p\|; [f_2 p^* + f_3 p^{\perp_1}, f_2 p^* + f_3 p^{\perp_2}, f_2 p^* + f_3 p^{\perp_3}, f_2 p^* + f_3 p^{\perp_4}]\big) \tag{16}$$

$$\text{with } p^* = X_{rot3}(\_, [a, \alpha]^T) \cdot p,\ p^{\perp_*} = \pm e_1^\perp \pm e_2^\perp$$
$$\text{and } f_1 = \tfrac{1-\cos\beta}{2},\ f_2 = 1 - f_1,\ f_3 = \sqrt{2}\sin\beta.$$

with $[x, y, z]^T = p^*$, $e_1^\perp = \frac{\sqrt{x^2+y^2+z^2}}{\sqrt{y^2+z^2}}[0, -z, y]^T$ and $e_2^\perp = \frac{\sqrt{x^2+y^2+z^2}}{\sqrt{y^2+z^2}}[y^2 + z^2, -xy, -xz]^T$ being an orthogonal set[3] of vectors having length $\|p^*\|$. The uncertainty β might as well be a parameter instead of being part of the configuration vector.

4) Location Joint: A 3-DOF joint for arbitrary 3d translation. Similar to the orientation joint it can incorporate relative position measurements with a single valued error bound $\triangle$, i.e. a sphere $\mathcal{Q} = \{q' \mid \|q' - q\| < \triangle\}$ of possible true locations around the configuration $q = [x, y, z]^T$. Our operations uses 1 point

$$\mathrm{Trans3D}_1\,(\_, [x, y, z, \triangle], p) := \big(\triangle;\ [\,p + [x, y, z]^T\,]\big) \tag{17}$$

A combination of a location and an orientation joint allows for handling static as well as arbitrary tracked moving objects in the environment.

5) Vehicle Joint: The vehicle joint represents the braking trajectory of a vehicle; it does not model the vehicles location and orientation (nor the location and orientation uncertainty), which can be done using a location and a orientation joint. The vehicle joint operates in the x-y plane and leaves z coordinates untouched. The braking motion starts at the origin. For now, we also assume that it starts tangent to the x-axis; omni-directional vehicles will be considered later on. The vehicle joint's input are not v and ω but the signed braking distance s and corresponding change of orientation α (Fig. 3(d)), which originate from the vehicle's velocity (v, ω). This is comparable to providing revolute joints with joint angle bounds $[q_0, q_1]$ as opposed to $(q, \dot{q})$.

A general vehicle joint operation not utilizing any motion constraints arises when signed braking distance and change of orientation are bounded separately and all potential robot

[3]This $e^\perp$ choice is numerically best for $|x| \le |y| \le |z|$, otherwise roles should be swapped.

poses within these bounds are considered to be reachable

$$\mathcal{Q}_\times = \mathcal{Q}_s \times \mathcal{Q}_\alpha \tag{18}$$
$$\text{with } \mathcal{Q}_s = \begin{cases} [0,s] & s\geq 0 \\ [s,0] & s<0 \end{cases} \text{ and } \mathcal{Q}_\alpha = \begin{cases} [0,\alpha] & \alpha\geq 0 \\ [\alpha,0] & \alpha<0 \end{cases} .$$

Such a conservative operation can be implemented by a prismatic and a revolute joint

$$\text{Vehicle}_\times\left(\_, [s, \alpha]\right) := \text{Circ}_2\left([0,0,1]^T, [0,\alpha]\right) \circ \text{Trans}_2\left([1,0,0]^T, [0,s]\right) . \tag{19}$$

First, the prismatic joint covers the straight braking component $\mathcal{Q}_s$. Then, its output is rotated by a revolute joint covering the orientation change component $\mathcal{Q}_\alpha$. Notice, that the revolute joint's rotation centre is in the vehicles reference point at braking motion start.

6) Vehicle Joint with Circular Motion Constraint: For vehicles constrained to circular motion, we utilize that constraint to shrink the braking volumes constituted by a vehicle joint. This becomes possible if the circular motion constraint is preserved by the vehicle within the whole braking, e.g., in case of a vehicle that keeps its steering angle fixed within braking. This yields a circular braking trajectory, which is the intuitive braking trajectory for moving cars. This behaviour is equivalent to slowing down v and ω proportionally or can further be seen as keeping the instantaneous centre of rotation (ICR) fixed. The motion model for such vehicles is a circular or straight motion starting at the origin tangent to the x-axis

$$X_{veh}(\_, [s, \alpha]) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & s \operatorname{sinc}\frac{\alpha}{2} \cos\frac{\alpha}{2} \\ \sin\alpha & \cos\alpha & 0 & s \operatorname{sinc}\frac{\alpha}{2} \sin\frac{\alpha}{2} \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{20}$$
$$\text{with } \operatorname{sinc}\phi = \begin{cases} \frac{\sin\phi}{\phi} & \phi\neq 0 \\ 1 & \phi=0 \end{cases} .$$

The configuration $q = (s, \alpha)$ represents the braking trajectory of the vehicle's reference point, which is assumed to be the origin. s is the arc length and α the corresponding angle. This representation jointly models circular trajectories ($s \neq 0$, $\alpha \neq 0$) as well as straight trajectories ($s \neq 0$, $\alpha = 0$) and even turning on the spot ($s = 0$, $\alpha \neq 0$). It avoids singularities other parametrizations, e.g., those using the ICR, have.

As operation input $[s_0, s_1, \alpha_0, \alpha_1]$ we use intervals for s and α which bound the uncertainty of braking distance and angle. The motion is covered by considering $\lambda[s, \alpha]^T$ with $\lambda \in [0, 1]$ representing the whole circular arc of the reference point:

$$\mathcal{Q} = \left\{ \lambda \left[\begin{smallmatrix} s' \\ \alpha' \end{smallmatrix}\right] \;\middle|\; \lambda \in [0,1], s' \in [s_0, s_1], \alpha' \in [\alpha_0, \alpha_1] \right\} . \tag{21}$$

The following operation simulates the four bounding motions and covers non-linear effects in the radius (Fig. 3(c)). It uses $4 * L + 5$ generating points with L being a parameter that adjusts the number of approximation points per arc:

$$\text{Vehicle}_{circ}\left(L, [s_0, s_1, \alpha_0, \alpha_1], p\right) := \left(r;\; \left[p,\; [[p^*_{s_i,\alpha_j}, [V^l_{s_i,\alpha_j}]_{l=0}^{L-1}]_{i=0}^{1}]_{j=0}^{1}\right]\right) \tag{22}$$
$$\text{with } p^*_{s_i,\alpha_j} = X_{veh}(\_, [s_i, \alpha_j]) \cdot p$$
$$r = \tfrac{1}{6}\left(\tfrac{\alpha_1-\alpha_0}{2}\right)^2 \max\{|s_1|; |s_0|\} + \left(1-\cos\tfrac{\alpha_1-\alpha_0}{2}\right) \|p\|$$
$$\text{and } V^l_{s_i,\alpha_j} = X_{veh}(\_, [\tfrac{l\cdot s_i}{L}, \tfrac{l\cdot\alpha_j}{L}]) \cdot U_{s_i,\alpha_j}$$
$$U_{s_i,\alpha_j} = p + Q(\tfrac{\alpha}{L})\tfrac{1}{2}(X_{veh}(\_, [\tfrac{s_i}{L}, \tfrac{\alpha_j}{L}]) \cdot p - p)$$
$$Q(\alpha) = \begin{pmatrix} 1 & \tan\frac{\alpha}{2} \\ -\tan\frac{\alpha}{2} & 1 \end{pmatrix}$$

For a proof of correctness, further details, or a simplified implementation without uncertainty see [3].

7) Omnidirectional Vehicle Joint: Today, a lot of mobile robots provide omni-directional motion, in particular many service robots such as DLR's Justin. The previously introduced vehicle joint operations can easily be extended to cover braking motions into an arbitrary direction ϕ. In Vehicle$_\times$ simply the axes of the translative component $\text{Trans}_2\left([\cos\phi, \sin\phi, 0]^T, [0, s]\right)$ has to be replaced to point into relative direction ϕ. For Vehicle$_{circ}$ the model X_{veh} gets replaced by

$$X_{veh+}(\_, [\phi, s, \alpha]) = \begin{pmatrix} \cos\alpha & -\sin\alpha & 0 & s \operatorname{sinc}\frac{\alpha}{2} \cos(\phi+\frac{\alpha}{2}) \\ \sin\alpha & \cos\alpha & 0 & s \operatorname{sinc}\frac{\alpha}{2} \sin(\phi+\frac{\alpha}{2}) \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{23}$$

Operations that consider ϕ with uncertainty are also possible.

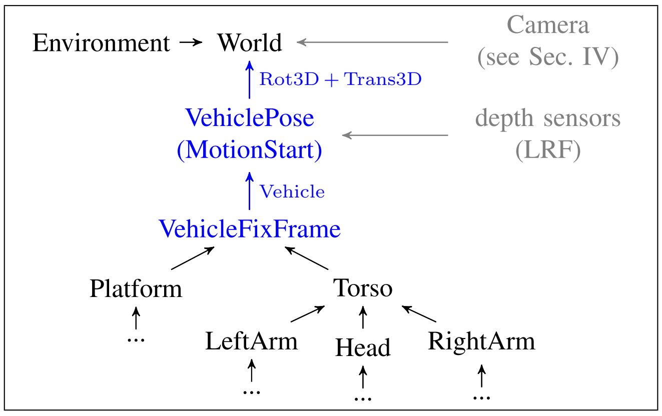

Fig. 4. Frames of Justin's kinematic tree. Blue frames model the vehicle motion. Locating the vehicle (Rot3D + Trans3D) is modelled separately from its braking behaviour (Vehicle). Gray elements show potential additional objects such as depth sensors for tracking or collision detection and the camera for the augmented reality visualization of Sec. IV.

D. Model for the Humanoid Robot Justin

Fig. 4 shows a kinematic tree for DLR's humanoid Justin including the environment and additional sensors that can be used for sensor based collision detection in the future. Using the world frame of the environment as the base frame allows to incorporate the environment into the self collision detection system proposed in this paper, but also

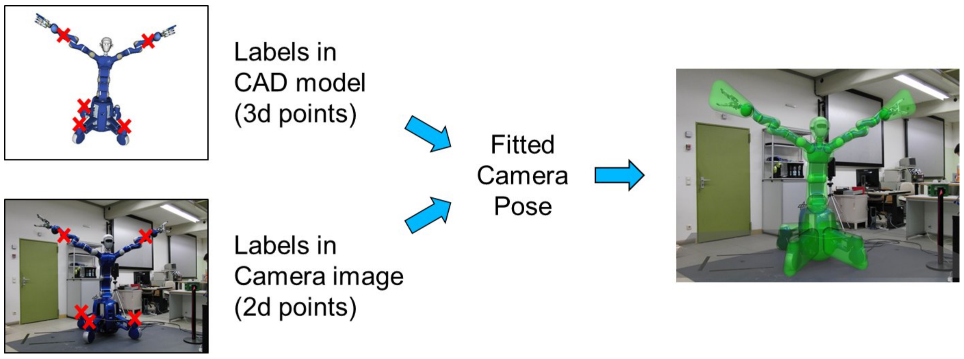
Labels in CAD model (3d points)
Labels in Camera image (2d points)
Fitted Camera Pose

Fig. 5. Workflow of camera pose determination for visualization of the collision volumes.

demands for providing robot pose measurements, which we did by dead reckoning from odometry. The blue elements in Fig. 4 cover the modelling of the vehicle motion. It shows a clear separation between locating the vehicle and its braking behaviour due to its current velocity. First, location and orientation joints provide the current vehicle pose, a frame that also is the start of the braking. Then, a vehicle joint models the braking from current velocity, and yields the vehicle fixed frame, a frame that is fixed to the robot and provides the base frame for the robots own kinematic structure.

Onboard obstacle sensors would provide there data in the VehiclePose frame. Consequently, our algorithm would operate in that frame and thus not include vehicle pose uncertainty in collision checks with the sensor data (future work), which is the intended behaviour for a local sensor. Nevertheless, the vehicle velocity would affect the swept volume size via the vehicle joint and thus braking distances would be included in the sensor based collision detection. Opposed to that, collision checks between a body and an (preconfigured) environment model would still be affected by braking distances and pose uncertainty.

IV. Augmented Reality Visualization

Besides configuration (see model construction in [2]) visualization is an important task in a collision avoidance system. Visualization of the collision model on top of images of the real robot should be used to validate the configuration data, i.e. the kinematic tree and bounding volumes. Visualization of the swept volumes may also support human as an assistive tool while operating equipment, e.g. a crane or lifting platform. We present a simple algorithm that allows to overlay volumes (SSCHs) or CAD models connected to nodes of the kinematic tree onto a series of images taken from a static camera in arbitrary pose. We use a simple but effective labelling process for determining the camera pose, thereafter SSCH are transferred into triangle-meshes as shown in [2] and finally visualized using the kinematic tree given the current configuration vector. The labelling has to be done once, the visualization is then possible for different configurations and camera images or even video data.

The workflow of camera pose determination is shown in Fig. 5. It starts by taking a picture of the robot in a fixed but unknown camera pose while the robot is in the known configuration q. The user now chooses arbitrary feature points and labels all of them twice: once in the camera image and once in a CAD model visualisation of the robot in configuration q. For the visualization we use Coin [20] in a QT4 GUI. The outcome of the labelling are 3d points from the CAD model returned in the base frame of the kinematic tree and according 2d observations of these feature points in the camera image. We then model-fit the camera pose where the projection of the 3d points matches best to the 2d points. For this model-fitting we use the MTK library [21]. The resulting pose links the camera to the kinematic tree allowing to transform every SSCH (or CAD model) into the camera frame for visualization.

For retrieving camera poses of good accuracy we suggest using a three stage process: (I) Adjust camera pose of the CAD model visualization roughly by hand (rotate and zoom using the mouse); (II) Label points and fit the camera pose (pre-labelling in bad camera pose); (III) Now use the fitted camera pose from (II) and label points again (labelling in good camera pose). Given the new labelling a refined fitting of the camera pose is performed.

Further, the labelled feature points should cover all dimensions of 3d space. Theoretically, a minimum of 3 labelled feature points is sufficient. Besides that, the resulting accuracy may depend on the image resolution and the accuracy of the kinematic tree resp. state q.

The features can also be labelled in different states q and corresponding camera images. If the camera is moving with a frame from the kinematic tree, this one must be used as base frame for the labelling.

V. Experiments and Results

Fig. 6 shows an evaluation of the camera pose error due to the labelling process. Instead of camera images we used images retrieved from our CAD model to simulate camera images. In this way, we obtained ground truth for the pose and removed all errors due to camera calibration, kinematic tree, or imprecise CAD models, thus showing the pure error of the labelling process. The results support our three step procedure (Sec. IV): It pays off to labels twice, so the final labelling is done in a perspective close to the camera view.

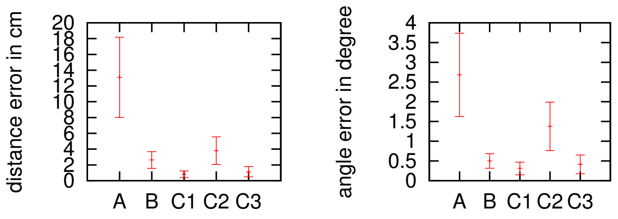

Fig. 6. Label accuracy: Error and standard deviation of camera position *(left)* and camera orientation *(right)*. Labelled feature points are: (A) 2 eyes, 2 hands in robot pose of Fig. 5. (B) 2 eyes, 2 hands, 4 wheels in robot pose of Fig. 5. (C) 2 mid points of torso joints (gray ellipse), 2 mid point of last arm joint (silver line) in robot pose of Fig. 7. The features in (A) are almost at the same depth leading to a poorly estimated camera pose. (C1-3) show how important it is to label the CAD model in a similar perspective as the camera. For comparison (C1) uses the true camera pose, i.e. ideally the same perspective with very good results. After step (II) (C2), i.e. labeling in a roughly aligned perspective the error is much worse. After step (III) (C3), i.e. again labeling using the perspective from step (II), the result is nearly as good as the ideal perspective in (C1). Feature points C were used for creating Fig. 7.

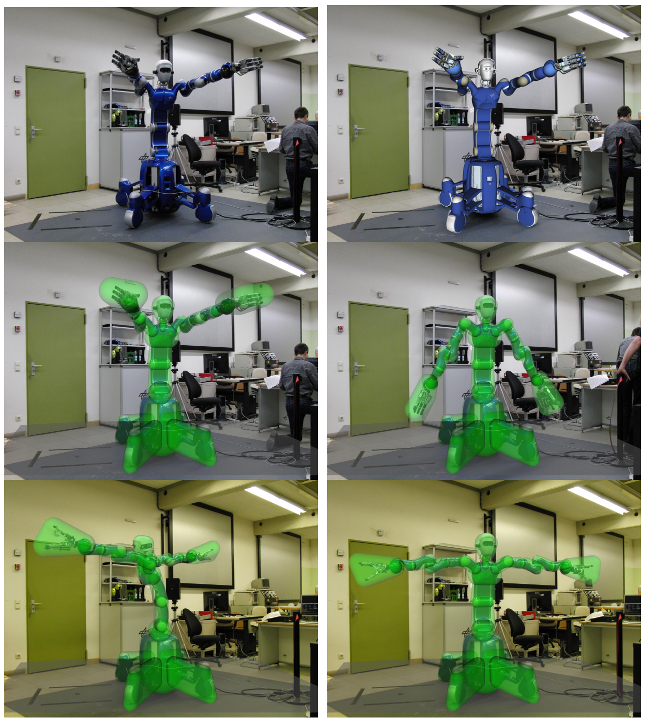

Fig. 7. Superimposition of Justin's CAD *(upper right)* and collision model *(middle and bottom)* over a real camera image *(upper left)*. The camera pose has been determined by labelling the upper left image using the labels described in experiment C.

Fig. 7 shows results of the visualization for real camera images of DLR's Justin. We used them to verify Justin's collision detection configuration. Images for multiple robot poses provide confidence in the correctness of the kinematic tree and the bounding volumes of the bodies.

Fig. 8 evaluates the computation times of the actual collision checking for DLR's Justin with and without its mobile platform.

Fig. 9 provides some examples of swept volumes due to combined motion of revolute and vehicle joints. A more extensive experiment is shown in the supplementary video, where 3d results are overlayed on a real video (Sec. IV) based on measured joint angles and odometry. In the video, scenes A show a single robot motion: first, overlayed with its resulting swept volumes; then, overlayed with the 3d CAD model. Scenes B show swept volumes (10 Circ_1, 10 Circ_2 joints, $\text{Vehicle}_\times$ joint type for the platform) for different motions of the platform (straight or spinning) with and without arm motion. It can be seen how the motions of the arm and the platform combine to the swept volume of the arm. This combination is the central point in an integrated vehicle-manipulator collision avoidance system. All swept volumes include a safety factor of 2, i.e. the deceleration is set to be only half the value the robot could actually perform – a reasonable assumption for real usage.

For the spinning motion it seems surprising that, while the platform is moving forward the swept volumes of the wheels point into different directions and even backwards for the rear wheels. This is indeed correct: The spinning motion is obtained by constantly changing the steering angle of all four wheels such that the ICR moves parallel to the robot center (in world coordinates) leading to a forward motion of the robot's center at any moment. The circular motion constraint in the vehicle joint model (Sec. III-C6) however assumes, that the wheel's steering angle and consequently the ICR are fixed during braking. In that case the robot would actually leave the spinning forward motion and continue in a tight curve as indicated by the swept volumes.

Scenes C physically validate the braking assumption and swept volume computation. At some point in time the robot starts braking and the swept volumes are fixed then, so one can see that the physical robot indeed stays inside the computed volumes. These scenes use no safety factor.

Finally, scenes D show the model validation process we proposed which indeed revealed a phenomenon we had not considered so far. D1 shows that when labelling the torso there are notable ($\approx$ 3cm) errors between the 3d model and reality (marked with arrows in the video). D2 shows that when labelling the platform, the platform itself fits well, but the robot has a similar error. It turned out that this is caused by the wheel suspensions that creates a small roll/tilt in the platform. To verify this, we have estimated the roll/tilt angle and as shown in scene D3 the result fits very precisely. Consequently, we incorporated the roll/tilt error into our collision model by adding an appropriate orientation joint that keeps orientation fixed but adds uncertainty as large as the observed upper bound of the roll/tilt angle. This leads to a larger swept volume but covers the suspension effect (E4). Note, that for checking the collision of two bodies the swept volumes in the least-common-ancestor frame are used [2]. Hence the additional uncertainty does not affect collision checks between parts of Justin's body but only between a body and an (preconfigured) environment model (see Fig. 4).

Due to the rendering process the overall computation time for the visualization is not constant but reaches frame rate (25Hz) for a complex model like Justin (INTEL i7@2.67GHz+NVIDIA NVS 3100M).

Model	Circ_1	Circ_2	Vehicle	FLOPS SV	time SV	time Dist	$\sum$ time
Fast Justin model	14	6	0	49.622	0.05ms	0.11ms	0.16ms
Fast Justin model	14	6	1	173.697	0.17ms	0.17ms	0.34ms
Accurate Justin model	10	10	0	130.496	0.12ms	0.27ms	0.39ms
Accurate Justin model	10	10	1	1.276.451	1.5ms	1.2ms	2.7ms

Fig. 8. Computational Effort: We tested collision models of DLR's Justin with and without the mobile platform. Configurations also differ in the number of joints that were modeled with the more accurate Circ_2. Cols 2-4 show the number of operations of each type, followed by the number of floating point operations and computation time for the swept volume computation. Last two col. show distance and overall computation time for one collision detection cycle. The distance computation starts each cycle from scratch until full convergence. It can be improved by reusing results from the previous cycle. From our experience the real-time technique from [2] used with an appropriate time budget allows roughly 3-times smaller distance computation times. Times were determined on a INTEL i7@2.67GHz.

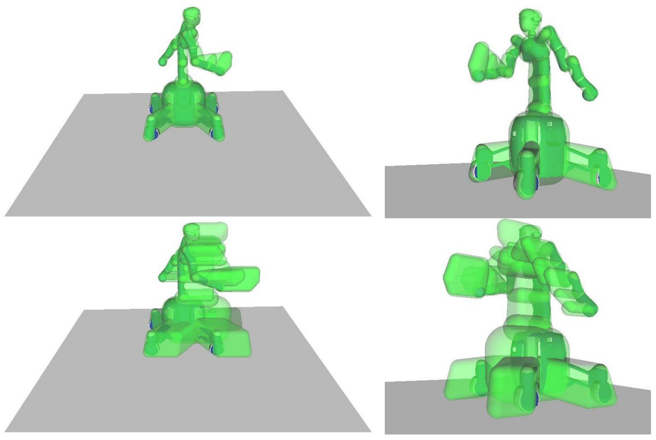

Fig. 9. Vehicle motion combined with motion of the right arm. *(top)* Two views of the arm motion. *(bottom)* Two views of the combined motion of arm and the mobile platform. Confer also supplementary video.

VI. Conclusion

We presented a general real-time continuous collision detection algorithm for arbitrary systems of moving bodies connected to each other in a kinematic tree of joints, in particular mobile manipulators and their environment. The approach is not restricted to a specific type of joints but generic with concrete formulas presented for revolute, prismatic, vehicle, location, and orientation joints. With the latter two as examples, joints even do not have to represent a physical connection but can represent a measured relative pose. We further introduced a augmented reality visualization of the collision volumes, an important tool for validating the collision systems safety. Future work will be the computation of the Jacobian of the distance results, an information needed for collision free path planning.

Acknowledgements

We thank Oliver Birbach for providing camera calibration.

References

[1] German Aerospace Center (DLR). Mobile Humanoid Rollin' Justin. http://www.dlr.de/rm/en/desktopdefault.aspx/tabid-5471/.

[2] H. Täubig, B. Bäuml, and U. Frese, "Real-time swept volume and distance computation for self collision detection," in *Proc. IEEE/RSJ Int. Conf. Intell. Rob. Syst.*, San Francisco, USA, 2011, pp. 1585–1592.

[3] H. Täubig, U. Frese, C. Hertzberg, C. Lüth, S. Mohr, E. Vorobev, and D. Walter, "Guaranteeing functional safety: design for provability and computer-aided verification," *Auton. Rob.*, vol. 32, no. 3, pp. 303–331, April 2012.

[4] C. Ericson, *Real-Time Collision Detection*, ser. The Morgan Kaufmann Series in Interactive 3D Technology. Morgan Kaufmann, 2005. [Online]. Available: http://www.sciencedirect.com/science/book/9781558607323

[5] G. van den Bergen, *Collision Detection in Interactive 3D Environments*, ser. The Morgan Kaufmann Series in Interactive 3D Technology. Morgan Kaufmann, 2003. [Online]. Available: http://www.sciencedirect.com/science/book/9781558608016

[6] E. Larsen, S. Gottschalk, M. C. Lin, and D. Manocha, "Fast distance queries with rectangular swept sphere volumes," in *Proc. IEEE Int. Conf. Rob. Autom.*, San Francisco, USA, 2000, pp. 3719–3726.

[7] S. A. Ehmann and M. C. Lin, "Accelerated proximity queries between convex polyhedra by multi-level voronoi marching," in *Proc. IEEE/RSJ Int. Conf. Intell. Rob. Syst.*, Takamatsu, Japan, 2000, pp. 2101–2106.

[8] B. Mirtich, "V-clip: fast and robust polyhedral collision detection," *ACM Trans. Graphics*, vol. 17, no. 3, pp. 177–208, July 1998.

[9] M. Tang, S. Curtis, S.-E. Yoon, and D. Manocha, "Interactive continuous collision detection between deformable models using connectivity-based culling," in *Proc. ACM Symp. Solid Phys. Model.*, Stony Brook, USA, 2008, pp. 25–36.

[10] C. Kim, C. T. Haas, K. A. Liapi, and C. H. Caldas, "Human-assisted obstacle avoidance system using 3d workspace modeling for construction equipment operation," *J. Comput. Civil Eng.*, vol. 20, no. 3, pp. 177–186, May/June 2006.

[11] S. Hwang, "Ultra-wide band technology experiments for real-time prevention of tower crane collisions," *Autom. Constr.*, vol. 22, pp. 545–553, March 2012.

[12] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Rob. Autom. Mag.*, vol. 4, no. 1, pp. 23–33, March 1997.

[13] J. Minguez and L. Montano, "Nearness diagram (ND) navigation: collision avoidance in troublesome scenarios," *IEEE Trans. Rob. Autom.*, vol. 20, no. 1, pp. 45–59, February 2004.

[14] A. Chakravarthy and D. Ghose, "Obstacle avoidance in a dynamic environment: a collision cone approach," *IEEE Trans. Syst. Man Cybern. Part A Syst. Humans*, vol. 28, no. 5, pp. 562–574, September 1998.

[15] ——, "Generalization of the collision cone approach for motion safety in 3-d environments," *Auton. Rob.*, vol. 32, no. 3, pp. 243–266, April 2012.

[16] P. Fiorini and Z. Shiller, "Motion planning in dynamic environments using the relative velocity paradigm," in *Proc. IEEE Int. Conf. Rob. Autom.*, Atlanta, USA, 1993, pp. 560–565.

[17] ——, "Motion planning in dynamic environments using velocity obstacles," *Int. J. Rob. Res.*, vol. 17, no. 7, pp. 760–772, July 1998.

[18] G. J. Hamlin, R. B. Kelley, and J. Tornero, "Efficient distance calculation using the spherically-extended polytope (s-tope) model," in *Proc. IEEE Int. Conf. Rob. Autom.*, Nice, France, 1992, pp. 2502–2507.

[19] P. G. Xavier, "Fast swept-volume distance for robust collision detection," in *Proc. IEEE Int. Conf. Rob. Autom.*, Albuquerque, USA, 1997, pp. 1162–1169.

[20] Coin3D library. http://www.coin3d.org.

[21] C. Hertzberg, R. Wagner, U. Frese, and L. Schröder, "Integrating generic sensor fusion algorithms with sound state representations through encapsulation of manifolds," *Inform. Fusion*, vol. (in press), 2012.